Universidad
Carlos III de Madrid
www.uc3m.es

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Sistemas Audiovisuales

TRABAJO DE FIN DE GRADO

Inferencia de la Respuesta Afectiva de los Espectadores de Un Video

Autor: Guillermo Mur Igualada

Tutor: Fernando Fernández Martínez

Madrid, 6 de Julio de 2015

RESUMEN

Con el crecimiento desmesurado de la cantidad de contenido audiovisual publicado a diario en internet se genera una cantidad de información que abre las puertas al análisis de dicho contenido de formas nunca antes posibles. El alumno de la Universidad Carlos III de Madrid, Alejandro Hernández García expuso, en su Trabajo de Fin de Grado *Aesthetics Assessment of Videos through Visual Descriptors and Automatic Polarity Annotation,* el empleo de características subjetivas junto con información objetiva para realizar un sistema capaz de predecir el sentimiento de un espectador hacia un video.

En este Trabajo de Fin de Grado se ha continuado dicha idea, obteniendo nuevas características de alto nivel, gracias a herramientas implementadas en MATLAB cuyo objetivo es la detección facial en videos. Esta investigación consta de 2 etapas: En la primera, se va a tratar de implementar las soluciones propuestas por MATLAB en una serie de videos que presentan condiciones muy adversas, y se va a extraer el mayor número posible de descriptores de ellos. En la segunda etapa, empleando la misma colección de videos que en el trabajo realizado por Alejandro, compuesta por anuncios de automóviles, y la información de clasificación extraída de ellos, se va a evaluar la utilidad de dichos descriptores a la hora de predecir la respuesta del usuario con respecto al video.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCTION

1.1 CONTEXT, MOTIVATION, AND OBJECTIVES.

In the new media age, video content creation and sharing has growth to never imagined levels. The main reason for this is the most powerful tool for information sharing ever created: internet. Thanks to internet, and a web in particular, YouTube, hundreds of millions of video content are uploaded and watched every day [25].

With this massive video traffic, along with the new social tendencies, there is an equal amount of information available about it, which greatly improves video tendencies prediction. However, most of this information is based on tags, comments, reviews, "likes", "shares", and many more, all of them based on the viewer's opinion.

This issue was addressed by Alejandro Hernández García in "Aesthetics Assessment of Videos through visual descriptors and automatic polarity annotation" [1], where he used aesthetic qualities, like colour or composition, extracted from a set of videos obtained from YouTube. Along with the users' opinion on these videos (obtained from YouTube metadata), he was able to create a model capable of predicting the user's sentiment towards a video.

His research proved to be a success, and opened a whole new field for investigation. This investigation was born as a new research for higher level features.

The starting point was a new MATLAB toolbox, introduced in r2014a, called Computer Vision System Toolbox [2]. This toolbox is a collection of algorithms, functions and apps for the design and simulation of computer video and video processing systems, such as object detection and tracking, feature detection and extraction, feature matching, or motion detection.

After doing some research, most of the tools provided were discarded. The reason for this, is that those tools were suited for very specific tasks, such as recognizing cars or people in security cameras, and they are not suited for any other kind of video. Others, like text recognition, provided little information about the video content.

However, there was a set of functions which proved to have great potential, those related to face detection and tracking. Using Viola-Jones detection algorithm and KLT or CAMShift tracking, these tools were originally implemented to find faces in ideal environments, such a webcam conversation, and wouldn't work in any other kind of video, we decided to try to bring out their capabilities, and adapt them in order to be used in any kind of video.

With this in mind, we developed a project planning with 2 main objectives:

- Create a MATLAB based program capable of detecting and tracking every face present on a video, regardless of their nature, and extract a set of features from them.
- Create a prediction model based on the results obtained by Alejandro in [1], and prove the use of these features measuring the impact on the viewer.

1.2 DOCUMENT STRUCTURE

In this section we will introduce the document's organization. It has been divided in several chapters, in order to provide a clear explanation of this research and its development.

- Chapter 1: An introduction to the context and motivations of this research, along with a brief description of the main objectives.
- Chapter 2: State of art analysis, applications, requirements, restrictions, and regulatory framework.
- Chapter 3: This chapter contains the development, both theoretical and practical, of the detection, tracking, and feature extraction software, developed in MATLAB.
- Chapter 4: Overview of features obtained with the program.
- Chapter 5: Explanation about the setup of the data mining software, WEKA, used for classification experiments.
- Chapter 6: Presentation and analysis of the results obtained by both the software implementation in MATLAB and the WEKA classification experiments.
- Chapter 7: Project Schedule and budget.
- Chapter 8: Main conclusions and future lines of work.

CAPÍTULO 2. PLANTEAMIENTO DEL PROBLEMA: ESTADO DEL ARTE.

2.1 ESTADO ACTUAL DE LA TECNOLOGÍA DE DETECCIÓN Y RECONOCIMIENTO FACIAL.

El campo de la detección facial presenta gran amplitud e historia, así como gran cantidad de aplicaciones distintas e implementaciones en función de las necesidades de las mismas.

La aplicación más extendida es el reconocimiento y clasificación facial, ampliamente extendida en el ámbito de seguridad, llegando a ser comparable a otras características biométricas, como huellas dactilares o reconocimiento de iris [26].

Aproximaciones más recientes a esta tecnología la sitúan en el campo del marketing orientado, implementando cámaras enfocadas al espectador. De esta imagen se obtienen características tales como género, raza, o edad, y con esta información mostrar publicidad de forma consecuente.

Estos avances traen consigo una gran dosis de controversia, ya que cuestiona la invasión de privacidad del espectador, como ha ocurrido recientemente con el sistema *OptimEyes* de la empresa Tesco [27].

Entrando en un plano más técnico, podemos distinguir dos etapas, detección y extracción de características faciales. En esta investigación nos vamos a centrar en la primera, ya que nos vamos a centrar en la influencia de dichas caras dentro del marco del video, y no en las características de la misma, pero vamos a dedicar una pequeña sección a evaluar los avances en dicha materia.

2.1.1 AVANCES EN DETECCIÓN FACIAL.

El primer paso es definir el concepto de detección facial como tal: La detección facial es un primer paso necesario en los sistemas de reconocimiento facial, con el propósito de extraer la región que contiene la cara del resto [28].

Un ejemplo de aplicación que empleamos día a día que contiene reconocimiento facial se encuentra en las cámaras digitales, la mayoría de las cuales incorporan este software con el propósito de realizar enfoque en las caras que se encuentran en la imagen.

El algoritmo más popular es el propuesto por Paul Viola y Michael J. Jones [4], el cual es el más empleado actualmente en tareas de detección facial, y en el que vamos a basar nuestra investigación (este será explicado en profundidad en el capítulo 4). Dos alternativas a este sistema de detección incluyen los propuestos por H. Rowley *Neural Network-based face detection* [29] o el sistema Bayesiano propuesto por Schneiderman y Kanade en [30]. Debido a su popularidad, podemos encontrar implementaciones en varios sistemas, como puede ser OpenCV o MATLAB.

El principal atractivo de esta implementación es una tasa de detección muy alta realizada en tiempo real. Este procesamiento en tiempo real no tiene por qué ser necesariamente a la misma tasa de imágenes que muestra el video, ya que para la mayoría de aplicaciones no es necesario obtener información de todos los frames. En la implementación descrita en [31] se describe la implementación en tiempo real como operación a 15 frames por segundo, que se ve reducida a 10 tras el proceso de reconocimiento.

Nuestro principal interés es la obtención de la mayor cantidad de información posible, por lo que será necesario renunciar completamente al procesado en tiempo real, con el fin de obtener una implementación más robusta, o al menos, que proporcione una cantidad de información suficiente en condiciones adversas.

Un buen resumen de dichas condiciones se puede encontrar en [32]:

- Posición: Las imágenes de una cara pueden variar debido a la posición relativa de la cámara con respecto a la cara (frontal, perfil, invertida), y algunas de las características en las que se basan estos sistemas puede verse parcial o totalmente ocluida como resultado.

- Presencia o ausencia de elementos estructurales: Características faciales tales como barbas, o accesorios como gafas pueden o pueden no estar presentes, además de presentar un alto grado de diversidad en factores como tamaño, forma o color.
- Expresión facial: La aparición de caras se ve directamente afectada por la expresión facial de la persona.
- Oclusiones: Las caras pueden verse parcialmente ocluidas por otros objetos. En imágenes que presentan grupos de personas, unas caras pueden ocluir parcialmente a otras.
- Orientación de la imagen: Las detecciones cambian significativamente en función de distintas rotaciones del eje óptico de la cámara.
- Condiciones de la imagen: Cuando la imagen se construye, factores como la iluminación o las características de la cámara afectan directamente a la aparición de las caras.

Como podremos observar en capítulos posteriores, nuestra colección de videos presenta todas estas condiciones, además de otras como puede ser la falta de resolución.

2.1.2 RECONOCIMIENTO FACIAL.

El campo de reconocimiento facial es mucho más amplio que el de la detección, además de no ser objeto de esta experimentación, por los motivos citados anteriormente.

Dicho esto, vamos a destacar una investigación reciente en este campo, en la que se asegura se superan los límites de la capacidad humana para esta tarea, obteniendo un porcentaje de acierto de 98.52% en el benchmark *Labeled Faces in the Wild (LFW)* mientras que el nivel humano alcanza un 97.53% [33].

2.2 INFLUENCIA DE LA APARICIÓN DE CARAS SOBRE EL ESPECTADOR.

En este apartado vamos a abordar el impacto que puede tener la aparición de una cara (y por extensión, de una persona) en un spot publicitario. Si bien este tema tiene más cabida en otros ámbitos como la psicología o el propio estudio de la publicidad, nuestra investigación trata de arrojar algo de claridad cuantificando dicha influencia, por lo que es necesario realizar una introducción acerca de este asunto.

La primera cuestión que surge es ¿Qué transmite al espectador la aparición de una cara en un video? La primera idea que surge es familiaridad, o la capacidad de sentirse identificado con el sujeto presente en el anuncio.

Los estudios más comunes de marketing hacen referencia al atractivo físico, y de cómo se relaciona este con el espectador [34], así como la personalidad adoptada por los actores [35].

También es posible encontrar investigaciones acerca de otros campos, como edad y género [36].

Nuestra idea inicial parte de olvidar dichas características de las personas y dar una medida del impacto que tiene la presencia de un rostro humano, independientemente de factores como género, edad o raza (si bien esto puede ser incorporado más adelante), una cuestión de la que no hemos sido capaces de encontrar documentación disponible.

Una rama de investigación reciente que está cobrando gran protagonismo es el neuromarketing. El neuromarketing se encarga de evaluar la respuesta cognitiva y afectiva de un espectador en respuesta a un estímulo producido por un anuncio [37]. Resultaría de gran interés para una futura investigación, unir los resultados aquí obtenidos con datos referidos a neuromarketing, pudiendo llegar a cuantificar los niveles de emoción y atención que presenta una persona frente a otra en un anuncio.

2.3 REQUISITOS.

A continuación se exponen los requisitos, básicos y opcionales, para llevar a cabo el siguiente proyecto:

- Equipo informático capaz de cumplir los requisitos mínimos de la versión correspondiente de Matlab R2014a (al tener este requisitos mínimos más altos que WEKA): Sistema operativo Windows XP en adelante, Mac OS X 10.7.4+ en adelante, o distribuciones cualificadas de Linux especificadas en la página web de Mathworks. Cualquier procesador Intel o AMD capaz de soportar el set de instrucciones SSE2, 2GB de memoria RAM, y espacio en el disco duro suficiente para contener Matlab con las herramientas adecuadas, más la colección de videos sobre la que trabajar (aproximadamente 20GB).
- Como ya hemos indicado anteriormente, el software necesario será WEKA, de libre distribución, y MATLAB, para el que necesitaremos una licencia.
- La colección de 138 videos sobre anuncios de automóviles, heredada del trabajo de fin de grado [1].
- (opcional). Equipo preparado para ejecutar tareas de gran coste de procesamiento, como puede ser el cluster del Departamento de Teoría de la señal y comunicaciones de la universidad Carlos III de Madrid.

2.4 RESTRICCIONES Y MARCO REGULADOR.

El software WEKA es de distribución libre y se puede utilizar sin restricciones.

En el caso de MATLAB, como ya hemos dicho antes, requeriremos una licencia original para la revisión R2014a en adelante.

Respecto al uso de videos comerciales, nos tenemos que remontar a su fuente, YouTube. Dichos videos han sido extraídos de YouTube, por lo que a la hora de emplearlos hay que hacerlo de acuerdo a su Términos y Condiciones de Servicio [3], concretamente, su uso con fines de investigación y nunca para un uso comercial.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN TÉCNICA.

En este capítulo se va a desarrollar la solución, tanto teórica como práctica, implementada en MATLAB.

3.1 DETECCIÓN

La tarea de detección se va a realizar mediante el algoritmo de Viola-Jones. Hemos escogido este debido a su popularidad, y a la existencia de métodos ya creados en MATLAB que van a facilitar su implementación.

3.1.1 DESARROLLO TEÓRICO DEL FRAMEWORK DE DETECCIÓN DE OBJETOS VIOLA-JONES

El algoritmo de Viola-Jones nace como resultado del framework de Viola-Jones para reconocimiento de objetos, siendo este la primera solución capaz de devolver una tasa de detecciones aceptables en tiempo real. Fue propuesto por Paul Viola y Michael Jones en 2001 [4]. Este framework se puede entrenar para reconocer gran variedad de objetos, pero la principal motivación de su creación fue la detección facial.

Sus principales características son una tasa de detección muy alta, con una tasa de falsos positivos muy baja, es capaz de trabajar en tiempo real (mínimo 2 frames por segundo), y únicamente es indicado para la detección facial, no el reconocimiento (es capaz de encontrar caras pero no de distinguirlas).

El algoritmo consta de 4 etapas:

La primera etapa consiste en la obtención de descriptores Haar-like. Estas deben su nombre a su similitud con el Wavelet de Haar, y ya habían sido empleadas en aplicaciones de detección facial, concretamente por C. Papageorgiou en 1998 [5]. La obtención de estas características se basa en las similitudes que comparten todas las caras, y que quedan reflejadas en imágenes con la diferencia en la intensidad de pixeles. Así pues, se pueden construir filtros rectangulares basados en estas diferencias, tal y como se puede ver en la siguiente imagen:

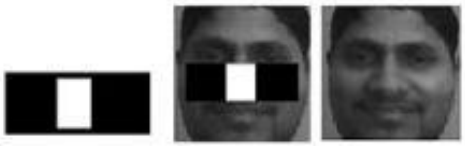

Figura 3.1: Descriptores Haar-like 1

En la imagen se puede observar una característica común: el puente de la nariz es más claro que los ojos. O en esta:

Figura 3.2: Descriptores Haar-like 2

En la que se aprecia claramente como la zona de los ojos es más oscura que las mejillas.

En la segunda etapa, mediante el uso de una representación de imágenes llamada Integral Image (más conocida como Summed area table [38] fuera del ámbito del procesado de imagen, se trata de una forma rápida y eficiente de obtener la suma de valores en un área rectangular), se evalúan estos descriptores de forma considerablemente más rápida frente a otras alternativas.

En la tercera etapa se emplea el algoritmo de aprendizaje AdaBoost para seleccionar las mejores características en un conjunto de imágenes y entrenar un clasificador, ya que de otra forma resultaría muy costoso (en una imagen de 24x24 pixeles hay un total de 162.336 posibles descriptores [6]). De esta forma se construye un clasificador duro como una combinación lineal de clasificadores blandos con pesos aplicados [7].

$$h(x) = sign(\sum_{j=1}^{M} \alpha_j \, h_j \, (x))$$

Cada uno de estos clasificadores blandos es una función umbral basada en un descriptor:

$$h_j(x) = \begin{cases} -s_j & if \ f_j < \theta_j \\ s_j & otherwise \end{cases}$$

El valor del umbral θ_j así como la polaridad $s_j \in \pm 1$ y los coeficientes α_j se determinan durante el entrenamiento del clasificador.

Una vez creado el clasificador llega al último paso, la arquitectura en cascada, la cual define el nombre de la clase empleada en Matlab `vision.CascadeObjectDetector`.

La idea de esta arquitectura es construir un árbol de clasificadores duros en forma de cascada, de forma que el primer clasificador, el cual emplea dos descriptores, obtiene un porcentaje de detección de casi el 100%, con un porcentaje de falsos positivos de entre 40% y 50% [4]. En el siguiente nivel se propone un mayor número de descriptores para filtrar ese porcentaje de falsos positivos. En el momento en el que un candidato no supera un nivel, este queda descartado automáticamente.

Con esta arquitectura se obtienen unos resultados muy buenos, aun teniendo unos resultados pobres en cada clasificador de forma individual: Una estructura de 32 niveles puede obtener una tasa de falso positivo de 10^{-6} con una tasa de clasificador individual de 65%. En el otro lado nos encontramos que para obtener una tasa de detección de aproximadamente 90%, cada clasificador individual debe tener una tasa de detección de 99.7%

Así pues, el algoritmo de Viola-Jones presenta las siguientes ventajas:

- Gran velocidad de procesamiento.
- Selección de descriptores eficiente.
- No depende de localización ni escala.
- Trabaja con descriptores, en lugar de la propia imagen.

A su vez, el algoritmo presenta las siguientes desventajas, las cuales se van a intentar corregir en este trabajo mediante procesamiento de video y empleo de otras técnicas.

- El detector solo tiene alta efectividad en imágenes de caras frontales.
- No es capaz de procesar caras que presenten un ángulo de rotación cercano a 45º tanto horizontal como verticalmente, debido a que trabaja con formas rectangulares.
- Es sensible a condiciones de iluminación.
- Puede obtener múltiples detecciones de la misma cara debido a la superposición de áreas a procesar.

3.1.2 IMPLEMENTACIÓN EN MATLAB DEL ALGORITMO VIOLA-JONES.

Como ya se ha expuesto en el apartado anterior, el algoritmo de Viola-Jones presenta una serie de puntos débiles innatos al propio algoritmo. A su vez, queremos utilizarlo como una herramienta que sea capaz de extraer información en videos comerciales, que se alejan mucho de las características ideales del detector, por lo que vamos a encontrar gran cantidad de falsos positivos y no-detecciones, entre otros problemas.

En primer lugar se implementa el algoritmo de Viola-Jones siguiendo para ello el ejemplo dado en la documentación de Mathworks para Matlab [8], esto es, empleando en primera instancia un método de búsqueda de posibles candidatos para caras frontales, y posteriormente una validación del mismo realizando una búsqueda del descriptor Haar para la nariz, visto en una imagen previa.

Este método se prueba como insuficiente en los videos tomados como ejemplo, ya que devuelve una cantidad alta de falsos positivos. Por ello, se llega a la conclusión de la necesidad de crear un sistema de "confianza" que nos ayude a facilitar la detección de un falso positivo. Para ello tomamos la idea de la arquitectura en cascada y emplearemos algunas de las características que nos facilita la clase `CascadeObjectDetector` de Matlab, asignándole un valor de 0 a 1 a cada cara en función de las características encontradas, siguiendo el siguiente diagrama:

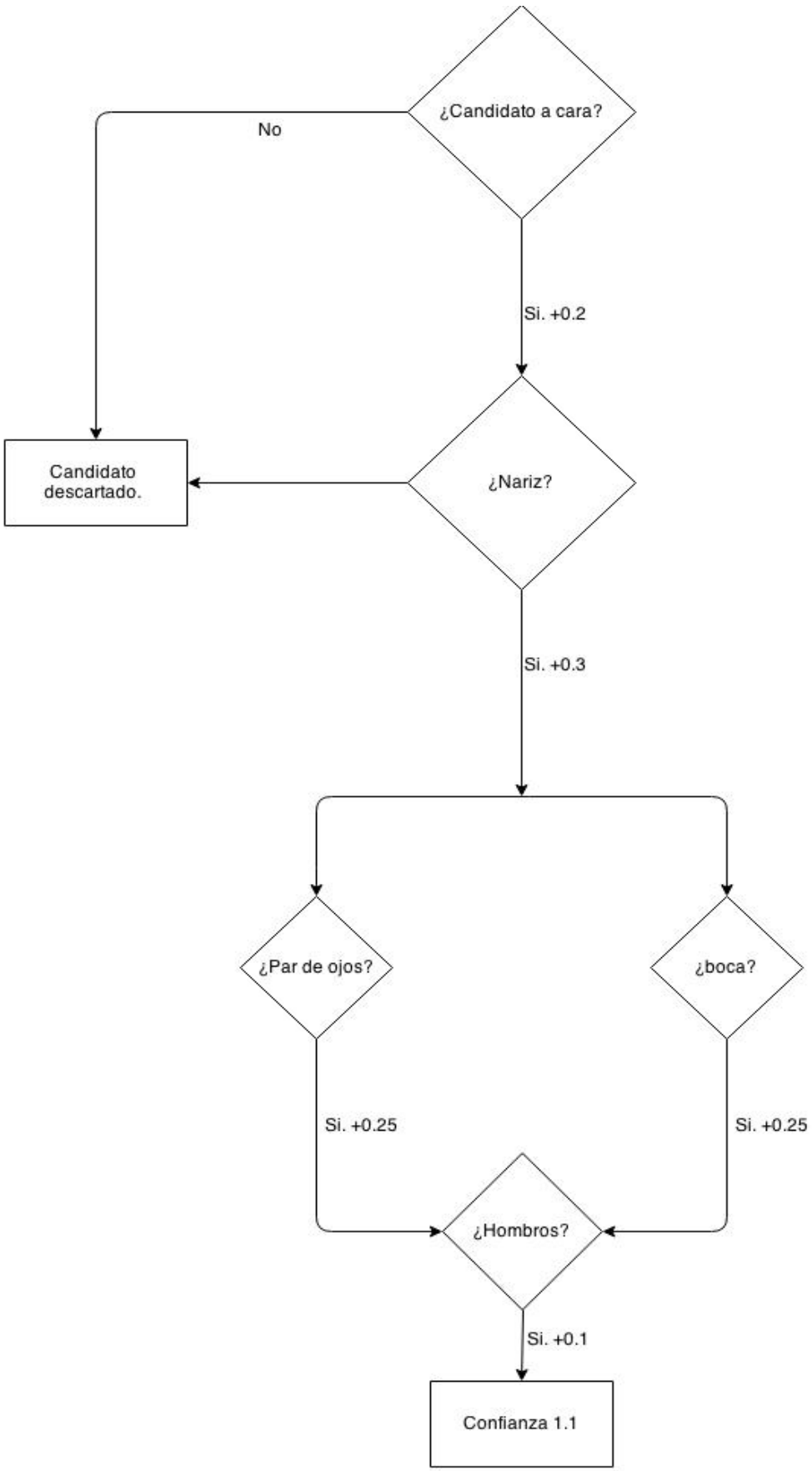

Figura 3.3 Diagrama de evaluación de confianza de detección 1

Como se puede apreciar la suma de todos los valores principales otorga un valor de confianza 1 a la cara. Si además es capaz de detectar la región superior del cuerpo (incluye hombros y cuello), este tomará un valor de 1.1, dando a entender que las posibilidades de que se trate de un falso positivo son despreciables.

Más adelante se incluye la posibilidad de que la cara se encuentre de perfil. Como ya hemos explicado anteriormente, el algoritmo de Viola-Jones es poco robusto a la hora de detectar caras de perfil, por lo que se decidió tratar este asunto empleando, una vez más, la medida de confianza para determinar la validez de la detección de dicha cara. En este caso, el diagrama se amplía de la siguiente forma:

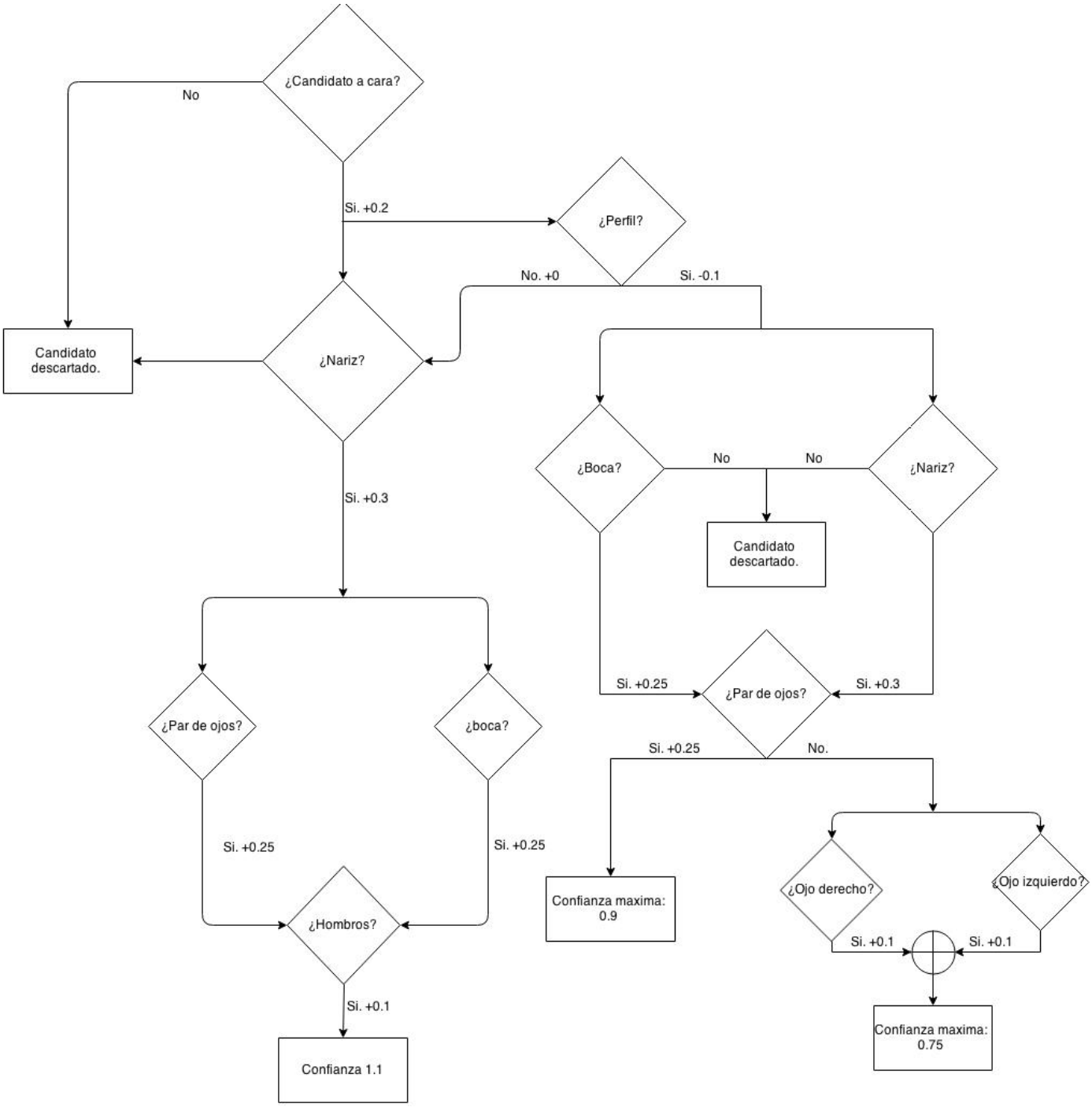

Figura 3.4: Diagrama de evaluación de confianza de detección 2

Salta a la vista que el valor de confianza máximo otorgado a caras de perfil es más bajo que el de las caras frontales. Esto es debido a la necesidad de hacer una distinción entre las caras frontales y de perfil (aparte de un flag específico para diferenciarlas en los resultados obtenidos), ya que estas últimas tienden a presentar una mayor tasa de falsos

positivos. Siguiendo con esta idea, se añadiría posteriormente un filtro que descarta cualquier detección posible con un valor de confianza inferior a 0.5, es decir, detecciones de caras de perfil que no cumplen con, al menos, 3 características del detector (el valor mínimo para detecciones frontales es de 0.5).

Una vez implementado el sistema de detección con una primera aproximación a un sistema para medir la confianza de la misma, procedemos a la segunda parte, el algoritmo de seguimiento.

3.2 KLT FEATURE TRACKER.

En los ejemplos facilitados por Matlab se proponen dos alternativas a la hora de realizar el seguimiento (o tracking) de las caras detectadas, CAMShift (Continuously Adaptative Mean Shift) y KLT (Kanade-Lucas-Tomasi) feature tracker.

El método de tracking CAMShift se basa en diferencias de histogramas entre la cara y el fondo para realizar el seguimiento de la misma. Este método presenta problemas debido a la dependencia de iluminación, y además nos aporta menos información que el algoritmo KLT, como vamos a ver a continuación, lo que nos hará decantarnos por este último.

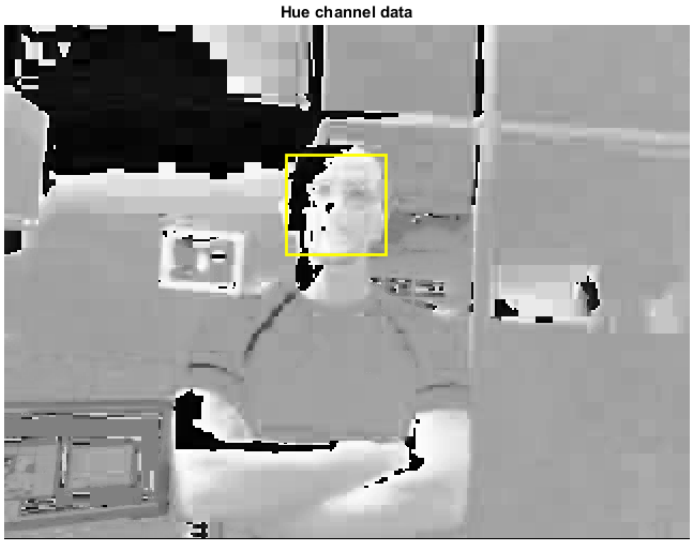

Figura 3.5 Ejemplo de tracking mediante CAMShift

3.2.1 DESARROLLO TEÓRICO DE KLT FEATURE TRACKER.

El algoritmo KLT nace como una alternativa de bajo coste a las técnicas tradicionales de seguimiento de imagen, debido a que evalúa una cantidad de coincidencias potenciales relativamente pequeña.

Inicialmente Lucas y Kanade [8] desarrollan la idea de una búsqueda local usando gradientes ponderados por una aproximación de la segunda derivada de la imagen.

Para el caso de una dimensión, encontramos h como el vector desplazamiento entre dos imágenes $F(x)$ y $G(x) = F(x+h)$, haciendo una aproximación

$$F'(x) \approx \frac{F(x+h) - F(x)}{h} = \frac{G(x) - F(x)}{h}$$

Obtenemos

$$h \approx \frac{G(x) - F(x)}{F'(x)}$$

Esta aproximación solo es correcta cuando el desplazamiento del área en concreto entre las dos imágenes no es muy grande. La aproximación de h depende de x. Combinando varias estimaciones de h para varios valores de x, se obtiene un promedio:

$$h \approx \frac{\sum_x \frac{G(x) - F(x)}{F'(x)}}{\sum_x 1}$$

Esta se puede mejorar ponderando la contribución de cada termino, la cual es inversamente proporcional a $|F''(x)|$, donde

$$F''(x) \approx \frac{G''(x) - F'(x)}{h}$$

Definiremos la función ponderada como:

$$w(x) = \frac{1}{|G'(x) - F'(x)|}$$

De forma que obtenemos el promediado ponderado:

$$h = \frac{\sum_x \frac{w(x)[G(x) - F(x)]}{F'(x)}}{\sum_x w(x)}$$

Este procedimiento se aplica repetidamente, de forma que al final la secuencia de estimaciones convergerá hacia el mejor valor de h. Estas iteraciones se pueden expresar como:

$$\begin{cases} h_0 = 0 \\ h_{k+1} = h_k + \dfrac{\sum_x \frac{w(x)[G(x) - F(x + h_k)]}{F'(x + h_k)}}{\sum_x w(x)} \end{cases}$$

Este procedimiento solo es posible en una dimensión, para realizar una aproximación en dos dimensiones hay que aplicar la siguiente transformación lineal:

$$F(x + h) \approx F(x) + hF'(x)$$

Para encontrar la h que minimice el error:

$$E = \sum_x [F(x + h) - G(x)]^2$$

Se aplica una derivación parcial y se iguala a 0:

$$\begin{aligned} 0 &= \frac{\partial E}{\partial h} \\ &\approx \frac{\partial}{\partial h} \sum_x [F(x) + hF'(x) - G(x)]^2 \\ &= \sum_x 2F'(x)[F(x) + hF'(x) - G(x)] \end{aligned}$$

Donde obtenemos el mismo caso que en una dimensión, salvo que la función de ponderación es $w(x) = F'(x)^2$ y su iteración se expresa como:

$$\begin{cases} h_0 = 0 \\ h_{k+1} = h_k + \dfrac{\sum_x w(x) F'(x + h_k)[G(x) - F(x + h_k)]}{\sum_x w(x)\, F'(x + h_k)^2} \end{cases}$$

Este método se puede extender (y en la implementación realizada en Matlab así se encuentra) para incluir transformaciones tales como rotación y escalado.

En segunda instancia, Tomasi y Kanade [9] emplearon el mismo método pero mejorado para identificar descriptores adecuados para un algoritmo de tracking. Estos descriptores serían seleccionados si ambos valores propios de la matriz de gradientes fueran mayores que un umbral establecido.

El problema se formula como $\nabla d = e$, siendo ∇ el gradiente y $\lambda_1\ y\ \lambda_2$ los valores propios que deben superar el umbral. Un método de tracking que implemente lo visto hasta aquí se considera un tracker KLT.

Una mejora adicional propuesta por Shi y Tomasi [10] consiste en una etapa más cuyo propósito es comprobar que estos descriptores se han seguido correctamente. Se crea una transformación afín entre la imagen que contiene el descriptor del cual se está realizando el tracking, y una imagen de un frame previo no consecutivo. Si la imagen afín compensada es muy distinta, el descriptor es descartado. La razón para esto es que entre frames consecutivos un desplazamiento es suficiente para realizar el tracking del objeto deseado, pero entre frames no consecutivos es necesario emplear un modelo más complejo.

Esto se puede explicar mediante la formula $Tz = a$, siendo T la matriz de gradientes, z el vector de coeficientes afines y a el vector error.

Una vez hemos explicado la base teórica de las dos funciones principales, queda aclarar que el principal motor del programa es el algoritmo de detección, siendo el de tracking un añadido que ayudara a resolver parcialmente algunos de los problemas expuestos.

3.2.2 IMPLEMENTACIÓN EN MATLAB DE KLT FEATURE TRACKER.

La implementación del algoritmo KLT, de la cual podemos encontrar una versión muy simplificada en [8], es algo más compleja que de CAMShift, si bien arroja unos resultados mucho más satisfactorios.

En primer lugar se detectan “eigenfeatures” (la extensión de eigenfaces a características faciales) para determinar una serie de puntos de interés en la bounding box que delimita el área donde se ha detectado una posible cara. En condiciones ideales, se devuelve una distribución de puntos que se centran mayoritariamente alrededor de la boca, nariz, y ojos.

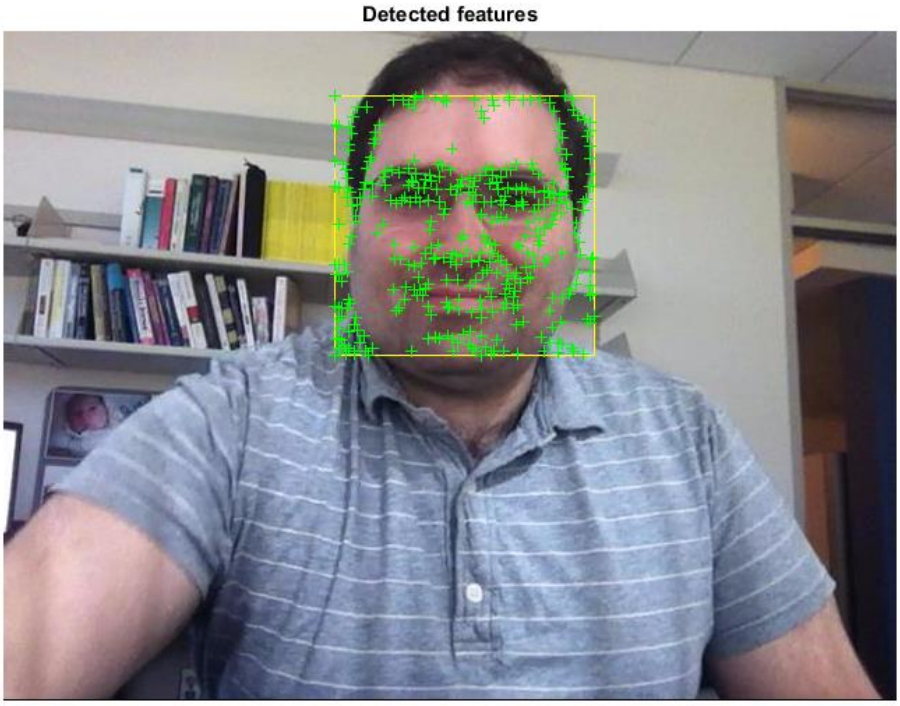

Figura 3.6 Representación de puntos de interés correcta

Sin embargo, al emplear videos en condiciones lejanas a ideales, obtendremos nubes de puntos sin una forma tan clara como el ejemplo anterior:

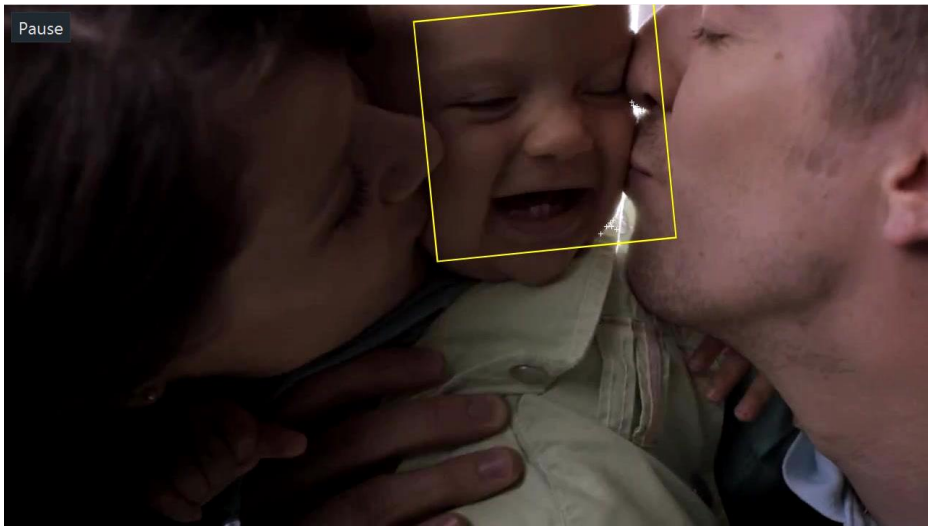

Figura 3.7: Representación de puntos de interés incorrecta

En esta imagen podemos apreciar una cara bien detectada, pero que a la hora de obtener los puntos de interés se realiza de forma incorrecta, debido a las condiciones de iluminación deficientes (los puntos se sitúan únicamente en la zona iluminada del borde de la cara).

Una vez tenemos la nube de puntos, podemos emplear la clase `vision.PointTracker` incluida en la colección de herramientas que estamos empleando. Esta se encarga de hacer el seguimiento de los puntos frame a frame, y además encerrar los mismos en una nueva bounding box. Hemos elegido este método por dos razones: La primera es, que a diferencia de realizar la detección frame a frame, esto nos permite identificar cada cara desde que aparece hasta que desaparece, lo cual nos va a permitir extraer información a priori importante, como puede ser posición y desplazamiento. La segunda es, que no solo nos permite realizar el seguimiento de las caras, sino que además, al ser una implementación de KLT, nos permite detectar el escalamiento y la rotación de la cara, solucionando casi en su totalidad uno de los mayores obstáculos que presenta la detección mediante Viola-Jones: mientras una cara no presente una rotación considerable en el primer frame en el que se marca como posible candidato, el programa es capaz de seguirla.

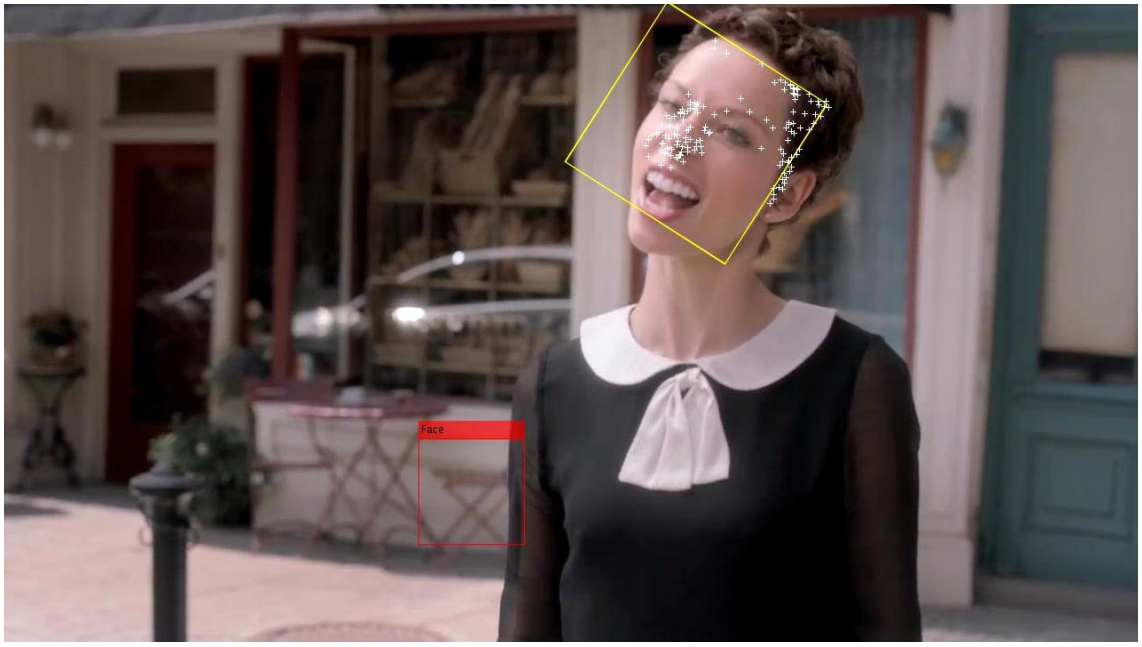

Figura 3.8: Cara con ángulo de rotación pronunciado

En esta imagen podemos apreciar una imagen con una cara rotada capaz de ser detectada gracias al algoritmo KLT.

Una debilidad encontrada en este algoritmo es su sensibilidad al cambio brusco de expresión facial, tal y como se puede observar en el siguiente ejemplo:

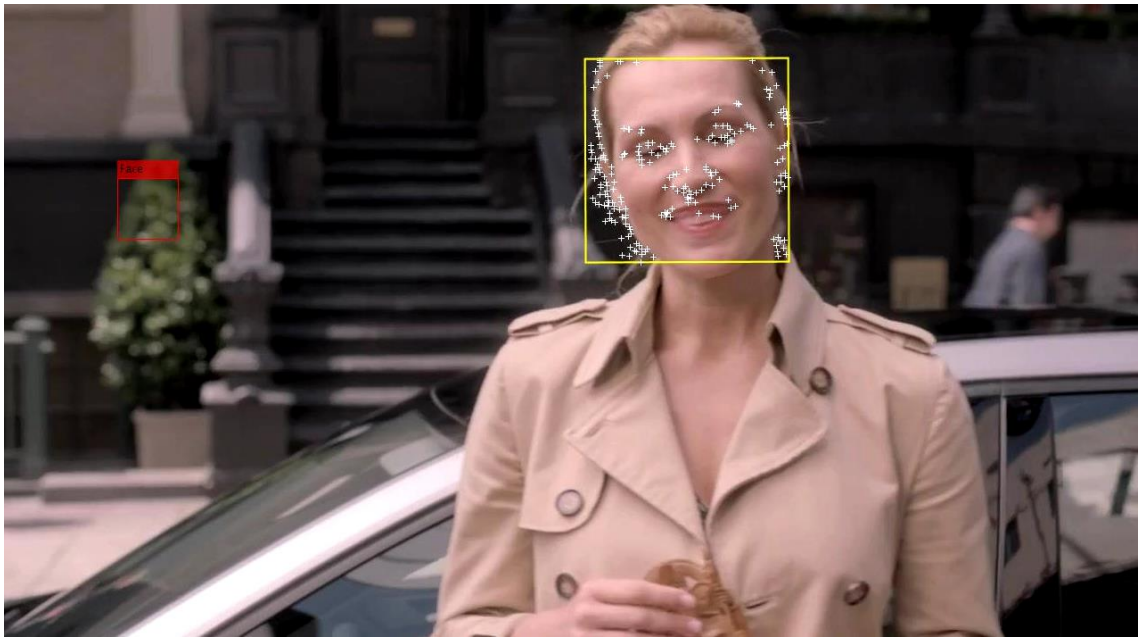

Figura 3.9: Muestra de puntos de interés en el frame de detección

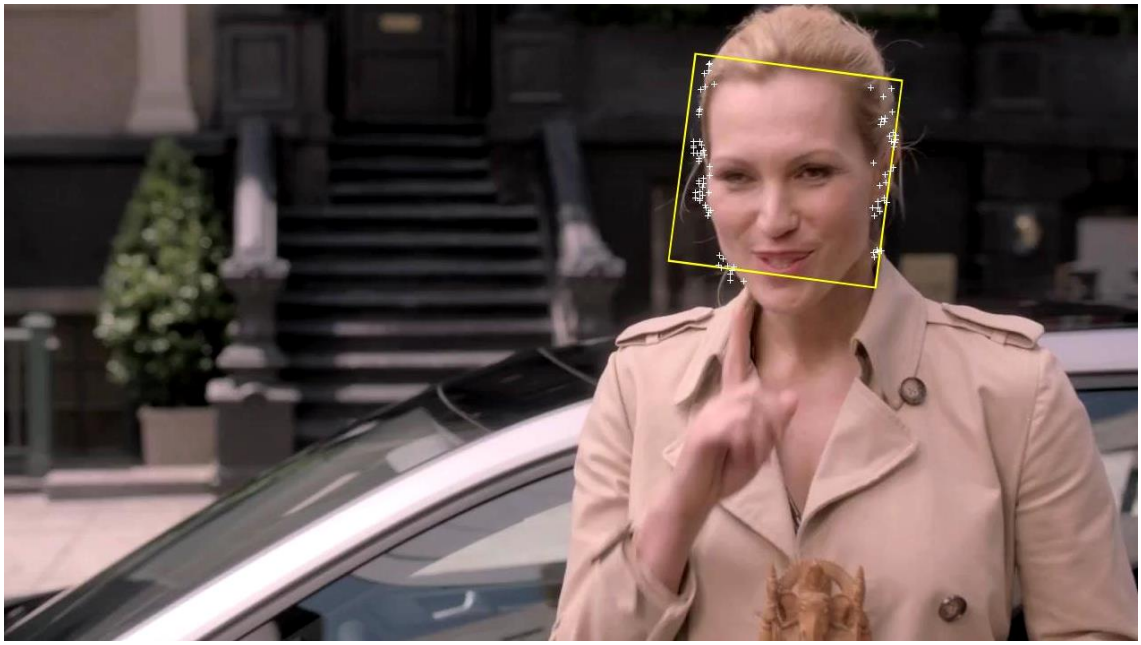

Figura 3.10: Perdida de puntos en 5 frames respecto a 3.9

Entre ambas imágenes hay 5 frames de diferencia, y podemos observar cómo se han perdido todos los puntos relativos a puntos de interés de la cara, quedando únicamente los relativos al contorno de la misma.

Este paso nos lleva a la creación de la estructura donde se va a almacenar la información de cada cara, y que en este momento presenta los siguientes valores: posición inicial y final de la cara, área que ocupa la misma en el momento de la detección, frame en el que es detectada y frame en el que se pierde, escena del video en la que se encuentra, y valor de confianza en la detección.

3.3 SEGUIMIENTO DE VARIAS CARAS Y DETECCIÓN SIMULTÁNEA.

Hasta aquí somos capaces de detectar una cara y seguirla, pero nos encontramos frente a dos nuevos problemas: el primero, es la necesidad de contemplar el seguimiento de varias caras simultaneas, y el segundo, poder detectar nuevas caras a la vez que se está realizando el seguimiento de las antiguas.

El primer problema que nos encontramos es que la implementación ofrecida por Matlab funciona únicamente para una sola cara en un frame. La solución de esto es sencilla:

encapsular los datos necesarios para el seguimiento entre frames en una estructura (puntos antiguos, actuales, visibles, y coordenadas de la bounding box) y analizarlas todas en cada frame.

El segundo problema se antoja más complejo. Si lanzamos la detección en un frame en el que ya se está siguiendo una cara, el detector va a volver a encontrarla, ya que se trata de dos procesos independientes. La solución resultó ser bastante sencilla. Se crea un stream de frames paralelo al que se escribe en el video, en el que se ocultan las áreas que comprenden las distintas bounding box de las caras ya reconocidas, de forma que el detector no las encuentre, y la información de caras nuevas encontradas en estos frames, en caso de que las hubiera, se dibujan en el stream principal, donde se está escribiendo el video.

Figura 3.11: Cara bloqueada para evitar detectarla en el siguiente frame

Como parte negativa a este método, encontramos un gran impacto en el rendimiento, ya que ejecutar el algoritmo de detección en cada frame es costoso (el tracking esta propuesto en un principio como un método para aliviar el coste computacional del programa, pero en última instancia es necesario emplear los dos métodos de forma simultánea en todos los frames para obtener un resultado satisfactorio).

En la siguiente imagen podemos ver cómo, efectivamente, a la vez que se realiza el seguimiento de una cara está buscando posibles nuevos candidatos (representados con rectángulos rojos):

Figura 3.12: Ejemplo de detección y tracking simultaneo

Se aprecia que sobre la cara ya reconocida no se está ejecutando el algoritmo de detección. También podemos observar como candidatos que en un principio deberían detectarse no lo hacen. Esto puede deberse tanto a la baja resolución del video como a la orientación de las caras, no siendo frontales sino en un ángulo de aproximadamente 45º, al margen de la que presenta una clara obstrucción.

Con esto podemos dar por cerrada la parte "mecánica" de este programa, es decir, que sea capaz de realizar funciones básicas de detección facial en videos comerciales. En el apartado "resultados y evaluación" se expondrán las principales virtudes y carencias de este sistema, así como posibles mejoras futuras.

A continuación se va a desarrollar el planteamiento de la segunda parte de este proyecto: la extracción de características del video referentes a caras empleando el programa desarrollado, primero explicándolas por separado, y luego dando una lista final de las que va a devolver el programa al finalizar su ejecución.

3.4 EXTRACCIÓN DE CARACTERÍSTICAS INMEDIATAS.

De forma muy sencilla empleando el código que ya tenemos se pueden extraer varias características notables acerca de las caras encontradas: área inicial y final, posición inicial y final, frame en el que es encontrada y frame en el que se pierde el tracking. Si bien por si mismas no pueden aportar mucha información, encontramos uso en ellas para definir características que si pueden ser de utilidad: el movimiento de la cara a lo largo de su aparición, tanto en los ejes horizontal y vertical, como en términos de profundidad (relevancia que tiene esta cara en la escena). También podemos observar su duración (medida en frames).

A estas medidas relacionadas con posicionamiento y tamaño en el frame se añade el ángulo de rotación que presenta la cara (si bien no es una medida que pueda tener tanto significado como las anteriores, será de utilidad para cálculos posteriores), la medida de confianza de detección de cada cara, si se trata de una cara frontal o de perfil, y la escena en la que aparece.

3.5 CÁLCULO DE CARACTERÍSTICAS EXPERIMENTALES: INTENTO DE REFORZAR LA MEDIDA DE CONFIANZA.

A continuación decidimos buscar características que si bien no son tan inmediatas de obtener, también pueden ser de utilidad.

La primera de ellas se relaciona con la regla de los tercios. Dicha regla se emplea en composición de imagen, de forma que esta se divide en 9 partes iguales, y los objetos relevantes deben posicionarse en las intersecciones que se crean:

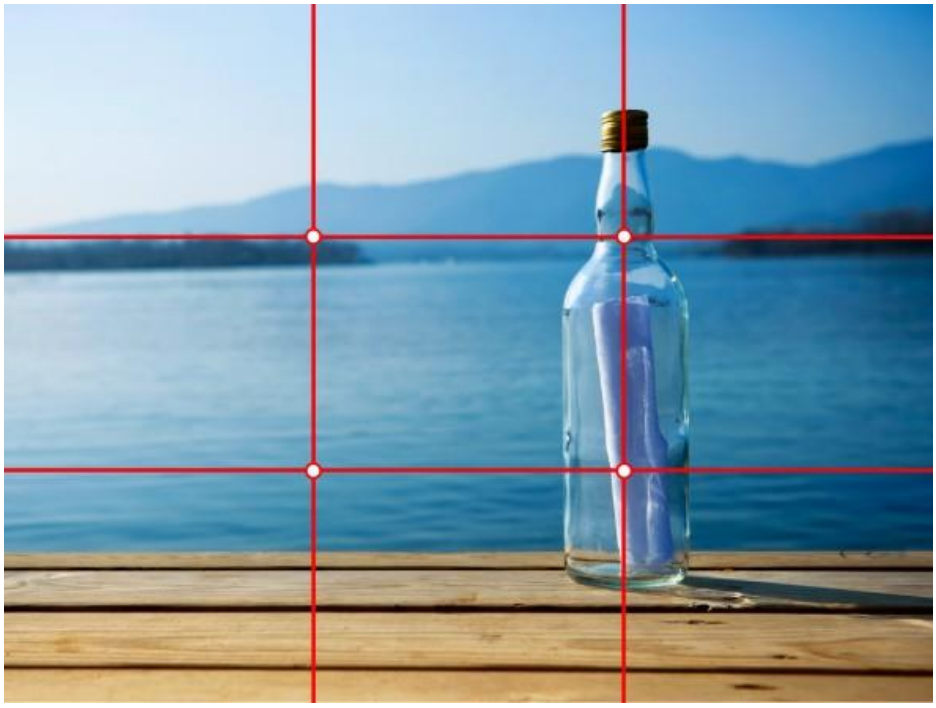

Figura 3.13: Empleo de regla de los tercios

Así pues, se desarrolla un sistema parecido al que ya hemos empleado para la medida de confianza de detección, que se estructura de la siguiente forma, tomando siempre como referencia que una cara se va a extender de forma vertical, y no horizontal:

- Si una de las intersecciones superiores está contenida en el área de una cara, la variable ROT = 1.

- Si una de las intersecciones inferiores está contenida en el área de una cara, ROT = 0.25.

- Si el área de una cara contiene dos intersecciones verticales (se sitúa sobre una de las divisiones verticales), ROT = 0.5.

- Para cualquier otra combinación, suponemos que es una cara que ocupa una gran extensión en el frame o que no contiene ninguna intersección, por lo tanto, ROT = 0.

Si bien es solo una medida aproximada, nos puede dar una idea de la posición que ocupan las caras en una composición basada en la regla de los tercios.

Por último, con el fin de reforzar la confianza en la detección, vamos a probar dos medidas experimentales, divergencia de Kullback-Leibler y distancia de Bhattacharyya.

Ambas medidas se apoyan en el mismo principio, la distribución de los puntos de interés localizados por el algoritmo KLT dentro de la bounding box, pero desde dos aproximaciones distintas. A continuación se realiza una breve explicación de ambas.

Divergencia de Kullback-Leibler: La divergencia de Kullback-Leibler (KL a partir de aquí) es una medida que se emplea para medir la diferencia entre dos funciones de distribución de probabilidad P y Q [13].

$$D_{KL}(P||Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)}$$

En una primera aproximación se definió una Q ideal uniforme dentro de la bounding box. Para P, dividimos la bounding box en una rejilla de 5x5 (se puede dividir en cuadriculas de mayor o menor tamaño, cambiando la precisión y la carga de procesamiento), y obtenemos la cantidad de puntos que se encuentran en cada división, para después normalizarla. Con estas dos medidas podemos calcular fácilmente la divergencia.

La idea tras este desarrollo es que una cara encontrada en buenas condiciones sigue una distribución bastante uniforme dentro de la bounding box, mientras que los falsos positivos suelen encontrar puntos de interés distribuidos de forma muy irregular, como el siguiente caso:

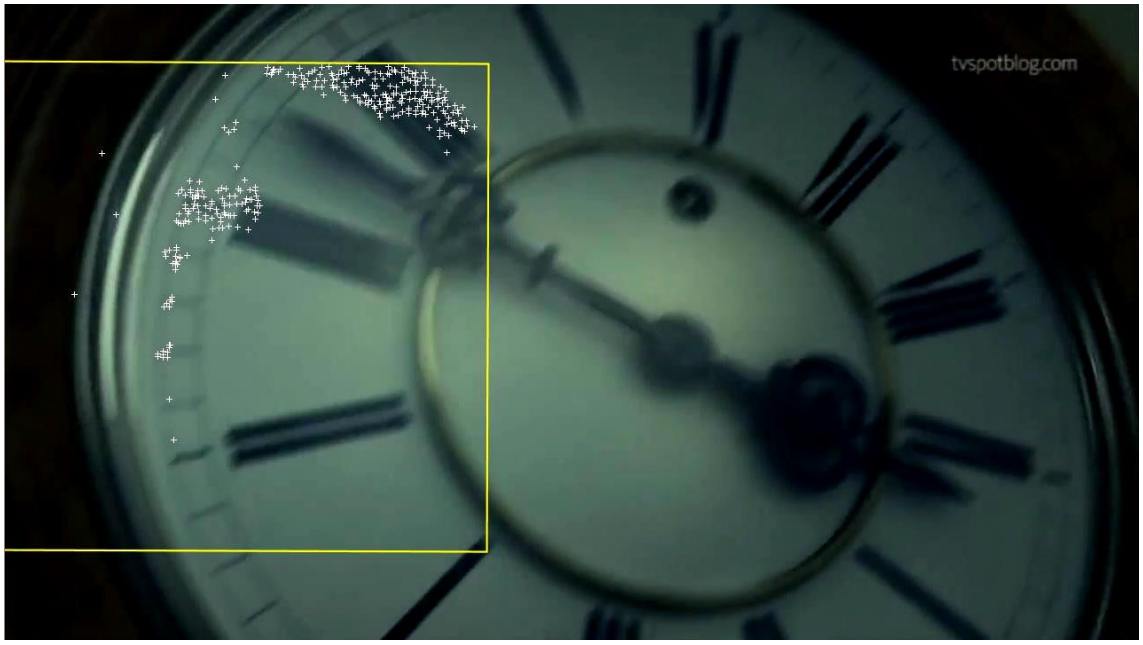

Figura 3.14: Distribución de puntos de interés en un falso positivo

Distancia de Bhattacharyya: Se utiliza para calcular la distancia entre dos distribuciones de probabilidad [14]. En concreto, utilizaremos el coeficiente de Bhattacharyya, despejándolo de la siguiente igualdad:

$$D_B(p,q) = -\ln(BC(p,q))$$

Siendo D_B la distancia de Bhattacharyya entre dos distribuciones normales, calculada como:

$$D_B = \frac{1}{8}(\mu_1 - \mu_2)^T \Sigma^{-1}(\mu_1 - \mu_2) + \frac{1}{2}\ln\left(\frac{det\Sigma}{\sqrt{det\Sigma_1 det\Sigma_2}}\right)$$

Siendo μ_i y Σ_i medias y covarianzas de las distribuciones, y

$$\Sigma = \frac{\Sigma_1 + \Sigma_2}{2}$$

El coeficiente de Bhattacharyya siempre toma valores entre 0 y 1. En primer lugar decidimos emplear distribuciones normales para poder medir como de centrada esta una distribución dentro de la bounding box, tomando una distribución gaussiana ideal centrada con una desviación típica igual a ¼ de la longitud de uno de sus lados (siempre son cuadradas).

Una segunda aproximación se realizó utilizando una rejilla similar a la utilizada en la divergencia KL, pero en lugar de ser uniforme, con forma de “cara”.

Debido a que estos cálculos se quieren emplear como medidas de confianza para detección y seguimiento, los emplearemos de la siguiente forma.

Por un lado, configuraremos unos umbrales de detección utilizando los valores de divergencia KL y coeficiente de Bhattacharyya, además del valor de confianza previamente calculado, que descartará automáticamente caras sospechosas de ser falso positivo.

Por otro lado, se empleará el coeficiente de Bhattacharyya (utilizando la máscara no uniforme) para crear una variable de confianza de tracking, calculada frame a frame.

Debido al alto grado de incertidumbre a la hora de realizar estas medidas, se hará especial hincapié en ellas en el apartado de experimentación y resultados, ya que más que ser características definitivas, se incluyen como un primer paso a realizar mayor experimentación.

Además se crean dos versiones del programa, una empleando umbrales y otra sin ellos, para poder comparar la eficacia de los mismos.

Para evitar posibles descartes erróneos de caras, se implementa el siguiente sistema. Si se localiza en una región dada cercana a una cara que ha sido marcada como falso positivo dentro de la misma escena, en una distancia no mayor a 5 frames, estas serán consideradas dos detecciones de la misma cara, y se unirán los datos de ambas.

Una vez realizados estos cálculos, estamos en disposición de extraer gran cantidad de descriptores que nos pueden ayudar a realizar la clasificación del conjunto de videos.

CAPÍTULO 4. EXTRACCIÓN DE DESCRIPTORES.

En este capítulo se van a detallar los descriptores extraídos de todo el proceso explicado en el capítulo 3. Puesto que el proceso de cálculo de estos es trivial, el capítulo se va a centrar en describir la posible aportación de cada uno, así como las decisiones tomadas a la hora de presentarlo.

De los valores que se ha creído podía ser útil, se ha tomado la media, desviación típica, y moda (discretizando los valores para obtener resultados significativos), debido al gran valor estadístico que presentan. En caso de no calcularse se especificará en la descripción del descriptor correspondiente.

Hemos dividido estos descriptores en 3 familias, en relación a su origen.

4.1 COMPOSICIÓN Y MOVIMIENTO.

La familia más evidente si partimos desde un punto de vista estético. Esta familia comprende:

- Área: En teoría una de las características más significativas, puesto que presenta relación directa con la importancia de la cara en la escena. Debido a las distintas resoluciones presentes en los videos, se normaliza el área calculada respecto al tamaño total en pixeles de la imagen, para dar uniformidad a la muestra. En concreto obtendremos el área media normalizada de las caras de cada video, su desviación típica, y la moda, discretizada en 5%, es decir, agrupar los valores de estas áreas en bins de 5% de ancho, y ver cuál es el más común.
  El valor de la desviación típica puede resultar interesante, ya que un valor alto significaría que hay un número similar de caras en el plano principal, y en un plano más secundario.
  De aquí también extraemos una variable binaria que indica si en el video hay alguna cara en primer plano, entendiendo como primer plano, que esta ocupe 1/3 del ancho y 1/3 del alto total del área del frame.

- Desplazamiento. A la hora de medir el desplazamiento, lo haremos tomando el valor absoluto del mismo para evitar obtener valores negativos, por dos motivos. El primero es que, a priori, es mucho más interesante conocer la "cantidad" de desplazamiento de una cara, es decir, cuanto se mueve durante su aparición, que la dirección de este. En segundo lugar, evaluar la dirección del movimiento anularía el valor de desplazamiento medio, ya que dos movimientos contrarios resultarían en un valor medio igual a 0. Hemos dividido el desplazamiento en eje X e Y, por un lado para comprobar si existe alguna diferencia significativa entre ambos movimientos, y por otro, para simplificar el procesado de los datos. De nuevo, debido a la diferencia de resoluciones entre los distintos videos, se han normalizado los resultados, está vez respecto al ancho y alto del video, para eje X e Y respectivamente. De nuevo se han obtenido media, desviación típica, y moda. También se ha obtenido una variable binaria que indica si hay desplazamiento en algún momento.

- Posición. Otro factor a tener en cuenta es la posición de las caras encontradas, ya que junto con el área son los dos descriptores que más información estética proporcionan de la cara en la escena. Al igual que el desplazamiento se ha normalizado con respecto al ancho y el alto. Obtenemos media, desviación típica, y moda.

- RoT. Si bien no es la forma más precisa de obtenerlo, la regla de los tercios tiene gran importancia a la hora de componer una escena, y debería darnos una idea aproximada de su relevancia en cada aparición.

- Ángulo de rotación. Posiblemente sea el descriptor menos significativo dentro de esta familia, pero decidimos incluirlo basándonos en la idea de que un movimiento de rotación muy pronunciado puede resultar poco natural y por tanto extraño para el espectador. En capítulos posteriores se valorará su utilidad.

4.2 DESCRIPTORES DE ALTO NIVEL.

En esta familia hemos incluido las características relativas al video como tal, en lugar de localizarlas a nivel de frame. La mayoría de características en esta familia son variables binarias o valores únicos para el video, por lo que no se podrá obtener media, desviación típica y moda.

- Número de caras. Quizá el descriptor más evidente de los que vamos a obtener, y también uno de los más relevantes, ya que es uno de los que mejor puede relacionar la presencia de caras en un video con la respuesta del espectador frente a estas, al margen de cómo se encuentren distribuidas a lo largo del video.

- Número de caras en la misma escena. Contabiliza solo escenas donde aparecen caras, con el fin de obtener algo más de información relevante a las escenas en las que aparecen caras. Siempre será igual o mayor a 1.

- Presencia de múltiples caras en un mismo plano. Variable binaria que indica si hay presencia de varias caras en un mismo frame, por lo que podemos afirmar casi con toda seguridad que existe interacción entre personas en al menos una escena.

- Porcentaje de frames que contienen caras: Nos da información acerca de qué porcentaje total del video presenta caras, por lo que, en teoría, debería tener un significado relevante.

- Porcentaje de escenas que contienen caras: Junto con el anterior, nos da una idea de cómo están distribuidas en el video, es decir, si hay muchas caras agrupadas en pocas escenas o si está más repartido.

- Número de caras por escena: En este caso tenemos en cuenta todas las escenas del video, para obtener información más precisa acerca de cómo están distribuidas el número de caras obtenido a lo largo del video. En este caso si podemos obtener media, desviación típica y moda.

- Duración de cada cara. Es necesario normalizar este valor al número de frames del video para obtener uniformidad en los resultados. Nos permite obtener media, desviación típica y moda, pero debería ser menos significativo que el

porcentaje total de frames que contienen caras, aunque aporta más información estadística.

4.3 MEDIDAS DE CONFIANZA.

En este conjunto englobaremos todas las medidas dedicadas a medir la confianza de la detección. En esta familia obtenemos media, desviación típica y moda para todos sus miembros, ya que al ser medidas de corte más experimental, es esencial obtener toda la información posible de ellas.

- Confianza de detección. El resultado de la clasificación hecha en base a la detección mediante Viola-Jones. Toma valores entre 0 y 1.1. En un principio debería ser la medida con más valor de esta familia, ya que es la que se apoya en la hipótesis más sólida.

- Confianza de tracking. Como ya hemos explicado antes, se trata de una medida experimental que se apoya en la distancia de Bhattacharyya empleando una distribución base modificada.

- Confianza global. Media calculada a partir de la confianza de detección y los distintos valores de la confianza de tracking.

- Distancia de Bhattacharyya. Distancia entre una distribución normal centrada en la bounding box y una aproximada a la distribución que siguen los puntos de interés. Se prestará especial atención a los resultados obtenidos para comprobar su utilidad.

- Divergencia de Kullback-Leibler: Utilizada para comprobar la uniformidad de la distribución, se encuentra en la misma situación que la distancia de Bhattacharyya.

Se van a evaluar todos los descriptores, tanto por familias como en un único grupo, esperando obtener mejor resultado de la segunda forma debido a la diversidad de las medidas obtenidas.

CAPÍTULO 5. PREPARACIÓN DE EXPERIMENTACIÓN. APRENDIZAJE MÁQUINA: WEKA.

Como ya hemos mencionado anteriormente, vamos a emplear la colección de 138 videos de comerciales sobre automóviles creada por Alejandro en [1], con el fin de dar continuidad al experimento.

El programa de Matlab se ejecuta de forma automatizada para cada video, generando un archivo .csv separado con comas que contiene los valores de los valores obtenidos para cada video. En un paso posterior, estos se unen en un único archivo, para poder utilizarlos en WEKA.

5.1 INTRODUCCIÓN.

La idea básica del aprendizaje máquina es la extracción de características de un set de datos, con el fin de crear unas reglas que puedan ser utilizadas posteriormente en otros conjuntos [15]. De esta forma se pueden crear programas que tomen decisiones sin que estas hayan sido explícitamente programadas [16].

WEKA es una colección de herramientas específicas de este campo, entre las que se incluye pre-procesado de datos, clasificación, regresión, clustering, reglas de asociación y visualización.

WEKA trabaja con archivos .arff, que pueden ser convertidos desde .csv directamente.

En nuestro trabajo emplearemos dos de las principales aplicaciones: *Explorer* y *Experimenter.*

5.2 WEKA EXPLORER

La primera aplicación que vamos a emplear es Weka explorer. Esta nos permite evaluar con detalle el contenido de un archivo .arff, dándonos información referente a cada descriptor con respecto a la clase (la cual nos indica si la valoración de un video es positiva o negativa, obtenida de la información de [1]).

Con esta herramienta vamos a separar en familias el archivo que tenemos con todos los valores obtenidos.
Además, vamos a emplear su herramienta de selección de atributos para crear subconjuntos de 5, 10, 15 y 20 atributos (en vista de los resultados que se obtengan esto puede ampliarse), con el fin de encontrar el punto en el que se produce overfitting y por lo tanto decae el porcentaje de acierto.

5.3 WEKA EXPERIMENTER.

A pesar de que la aplicación explorer tiene función de clasificador, este solo se puede utilizar con un descriptor, por lo que realizar pruebas consumiría demasiado tiempo.

Afortunadamente, WEKA cuenta con la herramienta Experimenter, que permite realizar la experimentación de forma mucho más rápida y presentando los datos de forma organizada.

Para los ajustes de dicha herramienta emplearemos validación cruzada de 10 iteraciones, distintos archivos con los conjuntos de características a analizar, y los siguientes clasificadores:

5.3.1 CLASIFICADORES UTILIZADOS.

En el experimentador de WEKA emplearemos los siguientes clasificadores. Se han seleccionado con el fin de emplear clasificadores de distintas clases, sin incurrir en excesos (la herramienta de WEKA contiene una gran colección de clasificadores disponibles).

- ZeroR: Se trata de un clasificador basado en reglas. Es el clasificador más simple, ya que únicamente depende de la clase, ignorando las predicciones. No tiene utilidad a la hora de definir predicciones, pero se puede utilizar como un benchmark para otros métodos de clasificación [19].

- NaiveBayes: Este clasificador bayesiano asume independencia entre las características a analizar, empleando el teorema de Bayes:

$$p(C|F_1, \dots, F_n) = \frac{p(C)p(F_1, \dots, F_n|C)}{p(F_1, \dots, F_n)}$$

  Siendo C la clase, F el descriptor, p(F|C) la probabilidad a posteriori, y p(C|F) la probabilidad de C en F [20].

- SimpleLogistic: Se trata de un clasificador mediante regresión logística. A diferencia de un clasificador lineal, este utiliza una curva logística que se define como

$$PR(Y = 1|X) = p(X) = \frac{e^{B0+B1*X}}{1 + e^{B0+B1*X}}$$

  En su implementación de esta basado en LogitBoost [21].

- Logistic. Este clasificador logístico mejora SimpleLogistic empleando un "ridge estimator" descrito por le Cessie en [22].

- SMO: Sequential minimal optimization (SMO) es un algoritmo creado para solucionar el problema de programación cuadrática (QP) que aparecer durante el entrenamiento de support vector machines (SVM). Fue creado por John Platt en 1998 [23].

CAPÍTULO 6. RESULTADOS.

6.1 RESULTADOS DEL DETECTOR.

Una vez finalizado el código, realizada una intensiva corrección de errores y fallos en casos particulares, se ejecuta el mismo sobre la colección de 138 videos. Cada uno de estos devuelve un archivo .csv con la información del video especificada anteriormente, y además genera un nuevo video, que presenta marcadas las áreas de posibles candidatos en rojo, y las detecciones positivas en amarillo, así como las distribuciones de puntos de interés de cada detección.

En un principio el código se iba a ejecutar sobre el cluster del departamento de Teoría de la señal y comunicaciones de la universidad Carlos III de Madrid, pero este no cumple el requisito de licencia de Matlab necesario, por lo que la única opción posible fue ejecutarlo en un ordenador de uso doméstico no preparado para tareas de gran coste computacional, lo cual ha dificultado en gran medida realizar una tarea más intensiva de experimentación, ya que cada ejecución tiene una duración en torno a 12 horas.

Una posible alternativa que se probó fue la utilización de la GPU del ordenador para realizar tareas de computación, ya que esta cuenta con núcleos CUDA (Compute Unified Device Architecture) [17], compatibles con MATLAB y con un rendimiento muy superior a la CPU, pero esta implementación no es compatible con Computer Vision System Toolbox.

También se evaluó la implementación de código para desarrollo en paralelo, empleando los métodos explicados en Parallel Computing toolbox [18], pero debido a la estructura del código, en la que el procesado de un frame depende del anterior, esta idea fue inmediatamente descartada.
Evaluar los resultados del detector se antoja complicado, ya que no se puede realizar de forma automática. Por ello, seleccionaremos 20 videos de los 138 disponibles y haremos un análisis de detecciones, para dar una idea general de la eficacia del mismo.

Video	Detect.	Falsos positivos	No det.	% falsas det.	% detección
Kia Cee'd	9	0	2	0%	81.81%
CLS	9	0	0	0%	100%
M. cuadros	7	0	0	0%	100%
M. Rayas	5	2	0	40%	100%
Citroen	9	2	6	22%	53.84%
Volks. Golf	8	1	3	12.5%	70%
Fiat 500L	7	1	4	14.28%	60%
Volks. Polo	29	4	8	13.79%	75.75%
Kia Sport.	5	1	2	20%	66.67%
Happy	25	1	12	4%	66.67%
Golf 2	10	3	6	30%	53.84%
Mercedes	13	1	10	7.69%	54.54%
Sandero	5	1	2	20%	66.67%
Hyundai	7	0	7	0%	50%
Outback	1	0	3	0%	25%
Kia Rio	7	3	4	42.85%	50%
Cit-2013	7	0	6	0%	53.86%
Peugeot	22	2	12	9.1%	68.75%
Merc. C	6	1	4	16.66%	55.55%
Ren. Clio	7	0	3	0%	70%
Total				**12.64%**	**66.14%**

Tabla 5.1: Evaluación de 20 videos procesados

Para obtener el total de caras presentes en un video, hemos seleccionado solo las que tienen relevancia en la escena, y que son visibles al espectador (no contamos aquellas difuminadas o en un plano muy lejano).

Si bien se trata de una muestra reducida, ya que el análisis de estos videos hay que hacerlo a mano, podemos extraer algunas conclusiones:

- El porcentaje de falsos positivos es relativamente bajo. La mayoría de estos se localizan en ropa u objetos tales como marcos de ventanas, es decir, en zonas que pueden cuadrar con descriptores Haar.
- El porcentaje de aciertos podría ser mejor. Vemos que hay un elevado número de no detecciones. Las causas más comunes de estas son:
    - Caras de perfil. Se trata de una de las debilidades del algoritmo de Viola-Jones, y a pesar de los intentos realizados para solventar este problema, no parece haber tenido un gran impacto, hemos obtenido algunas detecciones de caras de perfil, pero muchas se han obviado.
    - Obstrucciones. En el primer paso de la búsqueda de candidatos el algoritmo trata de localizar caras enteras. Si una cara se encuentra obstruida de alguna forma (esta tapada parcialmente por algún objeto en un plano más cercano, no aparece entera en el plano, o esta tapada por un complemento, como pueden ser gafas de sol), es muy probable que el algoritmo no la localice.
    - Baja calidad del video: Si el video presenta una resolución muy baja o una compresión muy alta, presentará una cantidad muy alta de no detecciones.

Una de las líneas de experimentación que se siguió durante un tiempo fue la creación de umbrales basados en distintos parámetros de confianza (se experimentó tanto con el valor de confianza de detección como con distintos valores para coeficiente de Bhattacharyya y divergencia de Kullback-Leibler), pero a la vista de estos resultados y de los efectos de estos umbrales (menor porcentaje de falsas detecciones, a costa de menos porcentaje de detección), decidimos descartarlo.

6.2 RESULTADOS DE WEKA.

La evaluación mediante WEKA se realizó tal y como se explica en el capítulo 5 de este documento. A continuación se presentan los resultados devueltos por WEKA:

Dataset	ZeroR	NaiveBayes	Logistic	SimpleLog	SMO	Regression
Top 5	55.77(2.33)	53.40(14.05)	62.08(11.67)	60.47(11.40)	58.65(7.57)	61.75(9.13)
Top 10	55.77(2.33)	55.68(15.36)	64.16(11.41)	63.75(11.45)	65.53(10.01)	63.95(11.18)
Top 15	55.77(2.33)	55.93(14.22)	60.36(10.72)	58.89(10.51)	64.72(10.18)	**66.05(9.94)**
Top 20	55.77(2.33)	55.17(13.64)	59.15(12.40)	56.29(9.46)	59.87(11.11)	62.60(10.94)
All	55.77(2.33)	49.49(14.14)	55.76(11.77)	49.90(10.56)	52.45(11.77)	51.46(11.70)

Tabla 5.2: Resultados de WEKA, entre paréntesis la desviación típica de cada valor.

En la tabla 5.2 se encuentran los experimentos realizados, con diferentes sets de datos, seleccionados mediante la herramienta correspondiente de WEKA para evaluar atributos empleando eliminación recursiva de atributos con SVM lineal (SVMAttributeEval) [24], y hemos creado 4 subconjuntos incluyendo los mejores 5, 10, 15, y 20 descriptores. Los mejores resultados se sitúan en torno a los subconjuntos de 10 y 15 atributos, con el pico en 15. La lista de estos atributos es:

- Desplazamiento_std_X
- KL_moda
- KL_std
- Rotación_media
- Desplazamiento_medio_x
- Posicion_moda_x
- Confianza_deteccion_media
- Desplazamiento_Std_Y

- Rotacion_media
- Bhattacharyya_media
- Numero_caras
- Confianza_deteccion_std
- Area_std
- Confianza_tracking_std
- Area_moda

De estos 15 descriptores, 8 se podrían considerar de tipo posición o movimiento, 6 pertenecen a las características que hemos definido como valores de confianza, y una relativa al número de caras, es decir, de alto nivel.

Tras realizar estas pruebas, vamos a proceder a realizar la misma, pero con los conjuntos de familias.

6.2.1 RESULTADOS POR FAMILIAS.

Tras repetir el experimento con los 3 conjuntos de datos por familias, obtenemos los siguientes resultados:

Dataset	ZeroR	NaiveBayes	Logistic	SimpleLog	SMO	Regression
Alt_niv	55.77(2.33)	48.26(12.98)	56.26(11.11)	53.66(8.92)	52.87(8.79)	55.25(10.12)
Comp.	55.77(2.33)	51.38(12.61)	53.61(12.75)	50.59(10.16)	53.29(11.19)	53.10(12.15)
Conf.	55.77(2.33)	48.51(11.24)	52.86(12.35)	52.31(10.17)	55.31(8.67)	55.36(9.79)
All	55.77(2.33)	49.49(14.14)	55.76(11.77)	49.90(10.56)	52.45(11.77)	51.46(11.70)

Tabla 5.3: Resultados de WEKA por familias.

Los resultados entran dentro de las previsiones, ya que al agrupar datos por tipo, estos van a compartir cálculos intermedios, y por lo tanto, va a haber menos diversidad, lo que se traduce en peores resultados en el proceso de clasificación.

6.3 CONCLUSIONES SOBRE LOS RESULTADOS DE WEKA.

De los resultados obtenidos en WEKA se extraer varias conclusiones.

- La primera y más importante, que hemos conseguido obtener un porcentaje de acierto en la clasificación con una diferencia destacable a la referencia ZeroR (máximo de 66.05% frente a 55.77%), por lo tanto la idea cumple con la premisa, y los descriptores obtenidos son de utilidad en el problema de clasificación de video expuesto.
- A pesar de haber obtenido 47 descriptores distintos, los mejores resultados se obtienen empleando solo 15. Esto se debe al problema de overfitting de aprendizaje máquina.
- Al agrupar resultados por familias obtenemos peores resultados, indicativo de que la diversidad en el origen de los datos obtenidos ayuda al problema de clasificación.
- A pesar de haber obtenido peores resultados empleando filtros en la detección, podemos observar que hay hasta 6 descriptores relacionados con los parámetros empleados para ejecutar estas técnicas de filtrado entre las 15 mejores, por lo que podemos concluir que estos son de utilidad, y el problema se encuentra en una mala implementación de los filtros.
- La aparición de ciertos descriptores en la lista de los seleccionados mediante SVMAttributeEval deja algunas conclusiones sorprendentes con respecto a las hipótesis iniciales, como puede ser la aparición de descriptores relacionados con rotación, la cual se pensó inicialmente como una característica más “débil” frente a otras presentes en su familia, o el gran número de descriptores de confianza que encontramos, probando que la experimentación con estos ha sido de utilidad.

CAPÍTULO 7. GESTIÓN DEL PROYECTO.

En este capítulo se va a exponer el desarrollo del proyecto, así como el número de horas empleado y un presupuesto estimado para el mismo.

7.1 ORGANIZACIÓN.

El proyecto se ha dividido en varias etapas. Debido a la idea que se quiere llevar a cabo, es necesario emplear un modelo en cascada, tal y como se muestra en el siguiente diagrama:

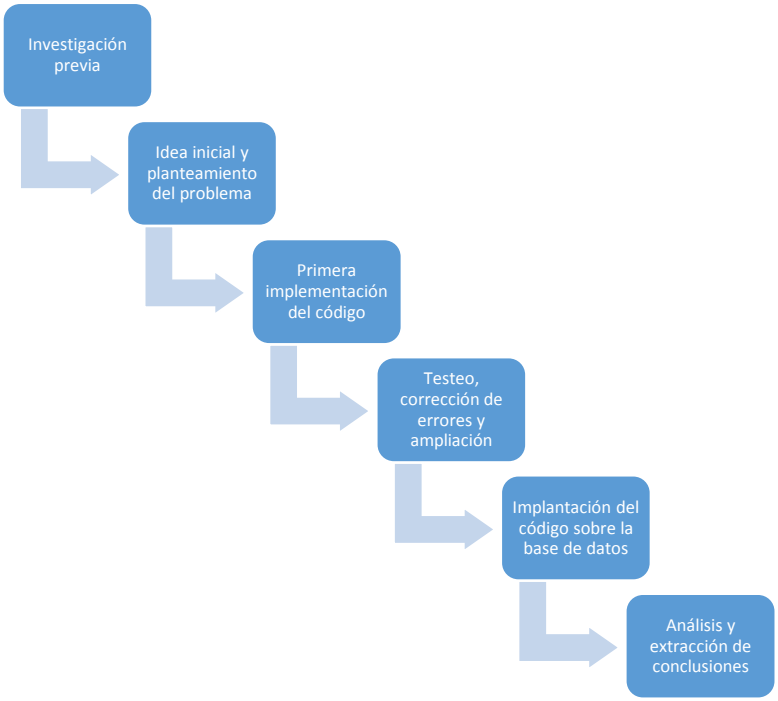

Figura 6.1 Diagrama de organización en cascada

A continuación se realiza una breve descripción de cada una de las etapas.

- Investigación previa: Todo el proceso inicial, anteriormente descrito, desde la decisión de buscar características de alto nivel hasta la decisión de enfocar el proyecto hacia la detección facial.

- Idea inicial y planteamiento del problema: Tras estudiar las herramientas básicas de las que disponíamos en MATLAB, se llevó a cabo un análisis de los requisitos realizó un primer diseño del código con las características que creímos necesario implementar.

- Primera implementación del código: Partiendo de la estructura ya disponible para análisis de frames y las herramientas de MATLAB se crea una primera versión del código, capaz de procesar videos completos y devolver una salida, tanto de video como de datos.

- Testeo, corrección de errores y ampliación: Se realiza experimentación con el código obtenido en videos aislados con el fin de encontrar errores en casos particulares y corregirlos, además de ampliar las funcionalidades del programa.

- Implantación del código sobre la base de datos: Una vez disponemos de una versión del código que creemos suficiente completa y estable, se ejecuta de forma automatizada sobre la colección disponible. Esta etapa tuvo que llevarse a cabo varias veces debido a la aparición de nuevos errores. En esta etapa descontamos la cantidad de horas de ejecución del código, ya que esta hacía imposible el uso de cualquier otra tarea.

- Análisis y extracción de conclusiones: Con los datos obtenidos de la ejecución se procede a evaluar la efectividad del proyecto, tanto en eficacia en la ejecución, como en utilidad de los resultados obtenidos.

A todas estas etapas hay que añadir la recolección de documentación y redacción de la memoria del proyecto.

En la siguiente tabla se muestra un desglose aproximado de la cantidad de horas empleadas en el proyecto:

Etapas	Horas empleadas
Investigación previa	20
Idea inicial y planteamiento	20
Primera implementación	50
Testeo, corrección y ampliación	80
Implementación	70
Análisis y extracción de conclusiones	30
Redacción de la memoria	70
Total	340

Tabla 6.1: Desglose de horas

7.2 PRESUPUESTO.

En este apartado se realiza una estimación del coste total del proyecto, tanto material como de personal.

El coste material se deriva tanto del equipo empleado para la realización del proyecto, como de las licencias empleadas. El coste del equipo es orientativo, y únicamente se van a tener en cuenta los componentes relevantes para el proyecto. Se incluye un desglose del precio, el periodo de amortización del mismo, y el coste en el periodo de duración del proyecto (aproximadamente 6 meses).

Componente	Precio	amortización	Coste
AMD FX-8370	201.53€	3 años	33.58€
SSD Samsung 850	142.71€	6 años	11.89€
Placa base Asus M5A99FX	141.5€	6 años	11.79€
Nvidia GTX 670	240 €	3 años	40€
PSU Corsair cx600	74.95€	6 años	6.24€
Caja de equipo	52€	6 años	4.33€
Total	**852.69€**	**-**	**107.83€**

Tabla 6.2: Costes de componentes

Para ejecutar el código se hace especial mención a un procesador de gama media-alta y al uso de un disco duro SSD, para facilitar el movimiento de grandes volúmenes de archivos de pequeño tamaño.

En un caso ideal se ejecutaría el código en un cluster de computación.

En cuanto al coste de las licencias empleadas:

Licencia	Precio	Amortización	Coste
MATLAB r2014a	79€	1 año	39.5€
Windows 8.1	-	-	-
Microsoft office 2013 Home	119€	5 años	11.9€
WEKA	-	-	-
Total	**198€**	**-**	**51.4€**

Tabla 6.3: Coste de licencias

El precio de la licencia de MATLAB es reducido debido a un descuento para estudiantes, en caso de no tenerlo incrementaría el precio. La licencia de Windows 8.1 es ofrecida por la universidad Carlos III de Madrid.

A continuación se detalla el coste de personal:

Puesto	Precio/hora	Horas	Coste total
Jefe de proyecto	25€/hora	50	1250€
Ingeniero	7€/hora	340	2380€
Total	**-**	**-**	**3630€**

Tabla 6.4: Coste de personal

Por lo tanto, el coste total del proyecto ascendería a:

Material	107.83€
Licencias	51.4€
Personal	3630€
Total	**3789.23€**

Tabla 6.5 Coste total del proyecto

CAPÍTULO 8. CONCLUSIONS AND FUTURE WORK.

In this chapter we're going to summarise the most important aspects of this research, along with some possible ways of improving the results.

8.1 CONCLUSIONS.

Following the path set by Alex in [1], we developed a program capable of extracting high level features from a video collection, in this case, related to face detection.

As we previously stated, the detector has an acceptable false detection rate, but the detection rate suffers from the weaknesses of the Viola-Jones algorithm, in which it's based, along with video related problems, such as low resolution, object obstructions, or lighting issues.

Despite of this fact, we managed to create a classification system based on 15 of these features, capable of obtaining a maximum classification accuracy of 66.05% in 2-class classification, which proves the point of this investigation. We believe this could be greatly improved with more time and/or better equipment, as it took a whole day to get a complete feature set which made checking improvements rather difficult.

Not every aspect of the investigation was a success, as we said before. The use of Bhattacharyya distance and Kullback-Leibler divergence as a threshold returned negative results. Despite of this fact, some of the features related to those measures made it into the top 15 features, which means they can be useful, and it is an idea that shouldn't be discarded yet.

Ideas exposed in chapter 6 serve as guidelines for future work in this task. Sadly we couldn't achieve to test everything we had in mind, but those ideas will be presented in the future work section.

In the end, this research has left two main contributions:

- The first one is the capability of using the Viola-Jones algorithm, together with KLT tracking, to extract several features related to face appearances in car advertisements. Proven the amount of negative conditions present in this collection, we believe that it should work in almost every other video, given reasonable limits. The code will be put at disposal for future works.

- The second one relies on the results given by WEKA. They can be seen as a first step in the quantification of the impact on the viewer of the presence of faces in a video. This kind of information could be very relevant in some fields, such as advertising.

8.2 FUTURE WORK.

We explained in 7.1 the strengths and weaknesses of our investigation, and now We're going to explain some possible lines of future research, as well as improvements:

- Improve detection rate. Detection rate presented by the program isn't terrible, but could be improved by developing alternatives to Viola-Jones algorithm used in specific cases, or create a more complex method using Viola-Jones algorithm, capable of getting higher detection rates, without getting higher false positive rates.

- Extracting more features based on facial detection. With the use of the points of interest returned by KLT algorithm, facial complexion through a video can be evaluated, and used to create more complex features.

- Consider alternative methods for face detection. As we stated in previous chapters, Viola-Jones algorithm is currently the most popular face detection system, due to a good compromise between detection rate and performance. That doesn't mean that it's the only one, and a further step in this investigation could be the addition of new face detection systems that improve the detection ratings. Our final goal is to achieve the highest possible face detection rate, despite its performance impact.

- Optimizing the code. In the reduced span of time we had to develop this project we did some basic optimization tasks, but this can be greatly improved.

- Extend the use of this program to other video categories. Although it was developed using car advertisements as a base, it should be capable of working in any kind of video category. Also, this would be positive in terms of proving the current feature's performance.

- Move towards face recognition technology. As explained in chapter 2, face recognition is quite an extensive field, and it could add some interesting information, like age, gender, race, or face emotion, which, in theory, should be highly related to the viewer's response.

- Right now we're relying on metadata from YouTube to evaluate if a video falls on the good or bad category, but a different approach to this evaluation process would have a great impact on proving the effectiveness of this investigation. We think that getting involved in neuromarketing investigation would provide very valuable data in order to improve the initial point of this investigation, knowing the viewer's response.

REFERENCIAS Y BIBLIOGRAFÍA.

[1] A. Martínez. *Aesthetics Assessment of Videos through Visual Descriptors and Automatic Polarity Annotation*. Universidad Carlos III de Madrid. 2014.

[2] Documentación de Mathworks referente a Computer Vision System Toolbox. http://es.mathworks.com/help/vision/index.html

[3] YouTube ToS. https://www.youtube.com/static?template=terms

[4] P. Viola, M.J. Jones. *Robust Real-time Object Detection*, IJCV 2001

[5] C. Papageorgiou, M. Oren, T. Poggio. *A General Framework for Object Detection*. International Conference on Computer Vision, 1998

[6] *Viola-Jones' face detection claims 1 Viola-Jones' face detection claims 180k features.* http://stackoverflow.com/questions/1707620/viola-jones-face-detection-claims-180k-features

[7] R. Szeliski. *Computer Vision, algorithms and applications*. Springer, 2010.

[8] *Face Detection and tracking using the KLT algorithm* http://es.mathworks.com/help/vision/examples/face-detection-and-tracking-using-the-klt-algorithm.html

[9] B.D. Lucas, T. Kanade. *An Iterative Image Registration Technique with an Application to Stereo Vision.* International Joint Conference on Artificial Intelligence, 1981.

[10] C. Tomasi, T. Kanade. *Detection and Tracking of Point Features*. Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.

[11] J. Shi, C. Tomasi. *Good Features to Track*. IEEE Conference on Computer Vision and Pattern Recognition, 1994

[12] B.F. Peterson. *Learning to see creatively*. Amphoto Press, 2003.

[13] S. Kullback, R.A. Leibler. *On information and sufficiency*. Annals of Mathematical Statistics 22, 1951: 79–86.

[14] A. Bhattacharyya. *On a measure of divergence between two statistical populations defined by their probability distributions*. Bulletin of the Calcutta Mathematical Society 35, 1943.

[15] R. Kohavi, F. Provost. *Glossary of terms*. Machine Learning 30, 1998.

[16] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer. 2006

[17] *Procesamiento paralelo CUDA* http://www.nvidia.es/object/cuda-parallel-computing-es.html

[18] *MATLAB Paralel computing toolbox* http://es.mathworks.com/products/parallel-computing/

[19] *ZeroR.* http://www.saedsayad.com/zeror.htm

[20] M. Narasimha Murty, V. Susheela Devi. *Pattern Recognition: An Algorithmic Approach.* 2011.

[21] N. Landwehr, M. Hall, E. Frank. Logistic *Model Trees*. Machine Learning 95(1-2) 2005.

[22] S. le Cessie, J.C. van Houwelingen. *Ridge Estimators in Logistic Regression*. Applied Statistics. 41. 1992

[23] J. Platt. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.* CiteSeerX. 1998

[24] I.H. Witten, E. Frank. *Data Mining. Practical Machine Learning Tools and Techniques*. Elsevier. 2005

[25] *YouTube Statistics.* https://www.youtube.com/yt/press/statistics.html

[26] *Facial Recognition Applications.* http://animetrics.com/technology/frapplications.html/

[27] *Tesco face detection sparks needless surveillance panic* http://www.theguardian.com/technology/2013/nov/11/tesco-face-detection-sparks-needless-surveillance-panic-facebook-fails-with-teens-do

[28] E. Hjelmas, B.K. Low. *Face Detection: A survey.* Computer Vision and Image Understanding 83, 236-274. 2001.

[29] H. Rowley, S. Baluja, and T. Kanade. *Neural network-based face detection*. IEEE Patt. Anal. Mach. Intell. 20, 22-38, 1998.

[30] H. Schneiderman, T. Kanade. *A statistical method for 3D object detection applied to faces and cars*. Computer Vision and Pattern Recognition, 2000.

[31] G. Shakhnarovich, P. Viola, B. Moghaddam. *A unified learning framework for Real Time Face Detection & Classification.* MItsubishi electric research laboratories. 2002.

[32] M.H. Yang, D.J. Kriegman, N. Ahuja. *Detectic Faces in Images: A survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 24. 2002.

[33] C. Lu, X. Tang. *Surpassing Human-Level Face Verification Performance on LFW with GaussianFace.* Cornell University. 2014.

[34] J.J. Watson, R.S. Rayner, S. Lysonski, S. Durvasula. *Vanity and Advertising: a Study of the Impact of Appearance-Related, Sex, and Achievement Appeals.* Advances in Consumer Research 26, 445-450. 1999

[35] A.M. Brumbaugh. *Physical Attractiveness and Personality in Advertising: More than just a Pretty Face?* Advances in Consumer Research 20, 159-164. 1993.

[36] M.M. Lee, B. Carpenter, L.S. Meyers. *Representations of older adults in television advertisements.* Journal of Aging Studies 21, 23-30. 2007.

[37] U.R. Karmarkar, *Note on Neuromarketing.* Harvard Business School. 2011.

[38] F.C. Crow. *Summed-Area Tables for Texture Mapping.* Computer Graphics 18. 1984